

# Accelerating Checkpoint-Restart with a Hardware Core

**Ashwin Mendon, Ron Sass**

{aamendon, rsass}@uncc.edu

Reconfigurable Computing Systems Lab

**Zachary Baker, Justin Tripp**

{zbaker, jtripp}@lanl.gov

Los Alamos National Lab



25 June 2012

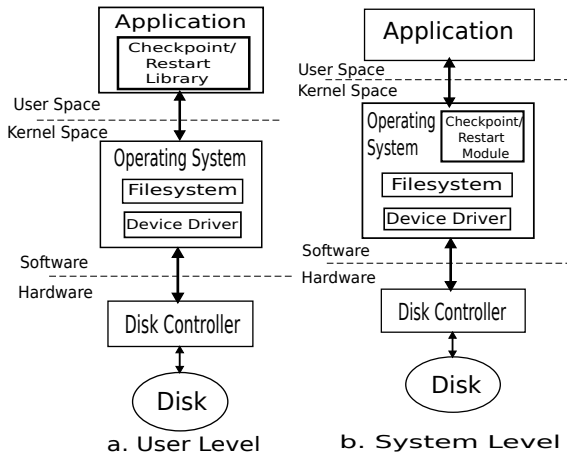
# Motivation

- System MTBF decreases with increasing number of cores and supporting components

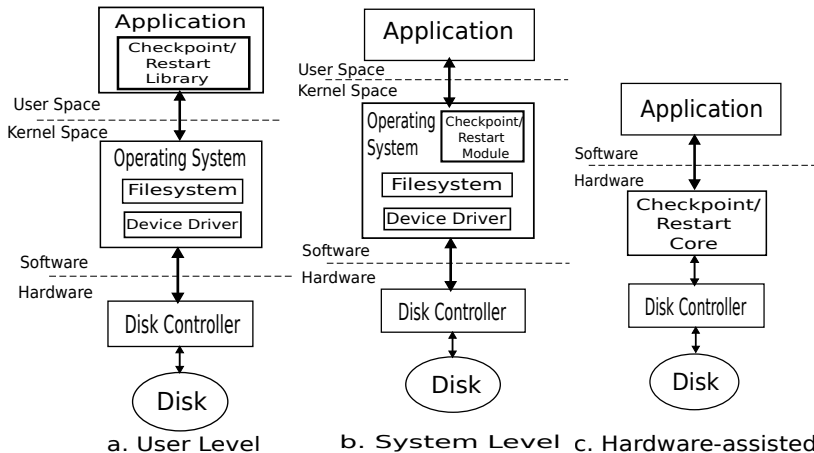
$$\Theta_{sys} = \frac{\Theta_{node}}{n} = \frac{5 \text{ yrs}}{100000} \approx 26 \text{ mins}$$

- Time to perform a checkpoint increases with application size
- I/O bandwidth to non-volatile storage limits checkpoint frequency

# Checkpoint/Restart Mechanisms



# Checkpoint/Restart Mechanisms



# Research Goals

## Key Question

Will a hardware-assist provide significant improvement over current approaches?

By *improvement*, we mean...

- Higher performance
  - Better SATA bandwidth
  - Lower overhead (compared to software)

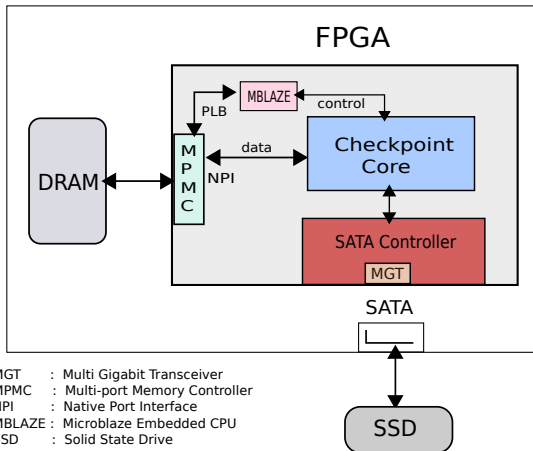
Ultimately, are checkpoints/restarts faster?

- Very modest resources

# Checkpoint/Restart Prototype

- Implemented prototype C/R core
- Implemented SATA2 disk controller core
- Integrated both on a FPGA that has access to node's memory

# C/R Block Diagram



Note: MBLAZE is a processor used to control experiments; it would not be part of the proposed approach.

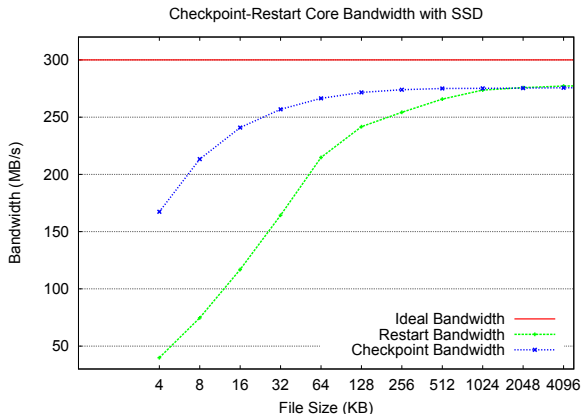
# Performance and Size

To evaluate the proposed solution

- Investigated bandwidth of various checkpoint sizes
- Measured checkpoint speeds versus raw SSD I/O
- Measured software overhead on conventional Linux node for reference
- Compare C/R core resources relative to a processor core

# Bandwidth

- Xilinx ML605 (V6 LX240T)
- OCZ Agility 2 SSD
- C/R and SATA instrumented to gather perf. data



# Overhead of C/R Core

<i>File Size</i> Bytes	<i>Checkpoint</i> time ( $\mu$ s)	<i>Restart</i> time ( $\mu$ s)	<i>SATA Write</i> time ( $\mu$ s)	<i>SATA Read</i> time ( $\mu$ s)
4096	24.48	102.76	24.35	102.67
8192	38.4	109.69	38.24	109.61
16384	68.01	140.18	67.9	140.12
32768	127.59	199.45	127.44	199.37

# Software Overhead

- SSD connected to an 2.1 GHz AMD Opteron Machine with 16 GB RAM
- `Fio`, a Linux benchmarking tool was used for raw read/writes

	<i>32 MB Seq Read time (ms)</i>	<i>32 MB Seq Write time (ms)</i>
CPU	155.07	331.25
FPGA	119.99	120.92

Overhead: 30% for Reads and 173% for Writes

# Software Effect on Bandwidth

	<i>Seq Read BW (MB/s)</i>	<i>Seq Write BW MB/s)</i>
CPU	206.35	96.6
FPGA	279.72	277.47

Bandwidth Lost:

- 26% for Reads
- 65% for Writes

# Resources Required

- V6 is 40 nm device with  $\approx$  2 billion transistors
- units: 150,720 LUTs, 301,440 Flip/Flops, 416 BRAMs, 20 transceivers

<i>Resources</i>	<i>CR</i>	<i>SATA</i>	<i>CR+SATA</i>	<i>% Device</i>
LUTs	1,280	1,531	2,811	1.8%
F/Fs	1,030	1,121	2,151	0.7%
BRAMs	4	3	7	1.6%
MGT	0	1	1	5%

# Conclusion

The propose C/R solution is...

- very small (fraction of a processor core),
- bandwidth efficient (approaches i/f speed),
- much faster than software.

These results suggest that the HPC community could benefit (if node-level storage was available).

# Future Work

- Presently, no filesystem (just sequential files)
- Might it be valuable to integrate with async/incremental system?
- How to implement....
  - Dark silicon on COTS, enabled for HPC machines?
  - Integrated SoC (ARM, BG node)?
  - As a peripheral card (maybe no SATA i/f?)

# Thank You

# Interface to Memory Controller and SATA HBA

